Deep Learning and Gradient Boosting Ensembles for Classification of Snake Species

Mirunalini Palaniappan[1], Karthik Desingu[1], Haricharan Bharathi[1], Eeswara Anvesh Chodisetty[1] and Anirudh Bhaskar[1]

[1]*Department of Computer Science and Engineering, Sri Sivasubramaniya Nadar College of Engineering, Chennai, India*

Abstract

This paper describes an deep-learning based ensembling approach for snake species classification. The proposed method employs state-of-the-art models such as ResNet and EfficientNet among others, applies transfer learning and fine-tunes them to the target data domain — snake images, uses them as feature extractors, and finally conjoins the produced representation vectors along with geographic metadata information to train a gradient boosting ensemble classifier to predict the snake species. The authors performed multiple experiments to train individual deep-learning architectures, select effective feature extraction models, and train a gradient boosting classifier using their ensembled features. The approach attained a maximum macro-averaged F1-Score of 51.39% on the test data. The corresponding validation F1-Score and Accuracy scores were 52.04% and 87.13%, respectively.

Keywords

Ensemble Learning, Convolutional Neural Networks, Gradient Boosting Ensemble, Feature Extraction, Metadata-aided Classification, Image Classification, Transfer Learning

1. Introduction

The SnakeCLEF challenge 2022 [1] held as a part of the LifeCLEF-2022 [2, 3] lab of the CLEF 2022 conference and the FGVC9 workshop organized in conjunction with CVPR 2022 conference aims to use image recognition to identify snake species.

The human population has been growing at an alarming rate from just over 1 billion in 1800 to 7.9 billion in 2000. This exponential increase in population has led to a demand for urban and rural dwellings, which has resulted in degradation of animal habitats. One of the species affected by this is snakes. India averages about 58000 snake bite deaths a year [4] which has become the result of frequent wildlife encounters. Snake bites can range from minor symptoms like nausea and breathlessness to major symptoms such as amputations and permanent disability. To address such problems, identifying the type of snake would help administer a precise antidote. Taxonomic identification of the species helps healthcare providers to articulate the symptoms, responses of the treatment and antivenom efficacy and also aids in clinical management [5].

Furthermore, snake species identification is crucial for biodiversity, conservation, and global health. Millions of snake bites occur globally every year, which often leads to snakebite

CLEF 2022: Conference and Labs of the Evaluation Forum, September 5–8, 2022, Bologna, Italy

miruna@ssn.edu.in (M. Palaniappan); karthik19047@cse.ssn.edu.in (K. Desingu); haricharan2010267@ssn.edu.in (H. Bharathi); eeswaraanvesh2010038@ssn.edu.in (E. A. Chodisetty); anirudh2010094@ssn.edu.in (A. Bhaskar)
0000-0001-6433-8842 (M. Palaniappan)

envenoming, killing, and disabling humans across the globe [6]. Because of the high intra-class and low inter-class variance, situational stress, and dread of potential danger, identifying the snake species from both a manual standpoint and a machine perspective is imperative. An accurate identification may also depend on various other factors, such as geographical location, morph, color, sex or age. Knowing the geographic location can also contribute more towards an accurate identification. An automatic system that helps in recognizing the snake species from the photographic image and geographic information can be paramount in overcoming the above problems. Hence, we propose an automated system that helps in identifying the snake species from the given input images.

The data provided in the task includes standardized images obtained from the online Biodiversity platform iNaturalist and photographed images. The dataset has a very long-tailed classification varying from about 6500 images to 5 samples. Different training set images such as large, medium and small sized images along with the metadata information were provided. We proposed a deep learning-based ensembling method using the large training set of images and the metadata information.

2. Related Work

Deep learning-based models such as EfficientNets and Vision Transformer (ViT) models were used and the prior probabilities of the location information were multiplied with the model predictions was proposed in [7] to classify the snake species. A deep learning architecture ResNeXt50-V2 was proposed for identifying 772 snake species [8]. Taxonomy based features were used in classification of snakes in [9] which has used similarity nearest neighbor classifier. The holistic methods such as k-nearest neighbors (kNN), support vector machine (SVM) and logistic regression (LR) are used in combination of dimension reduction approach PCA and LDA and the work was compared with CNN was proposed [10]. Study of deep learning, its strategies, comparison of frameworks, and algorithms were presented [11].

2.1. SnakeCLEF

The best scoring team's paper in SnakeCLEF-2021 written by Regő Borsodi [12], used an EfficientNetB0 base model while incorporating object detection and preprocessing using an EfficientNetD1 object detector. They got the highest F1 score of 0.90 with close to 95% classification accuracy. The 3rd best scoring team's paper in SnakeCLEF-2021 [13] written by Rail Chamidullin, went ahead with an ensemble of 4 models choosing to include ResNeSt50, ResNeSt101, ResNeSt200, and ResNet101. A majority voting strategy was employed ResNeSt200 was chosen since the model achieved an F1 score of 0.83 and a 91.60 classification accuracy.

The paper submitted by Lucia Georgiana Coca for SnakeCLEF-2021 [14] used 3 different models such as GoogleLeNet, VGG16 as well as ResNet. Vision transformers were used and an ensemble ResNet model were employed and achieved F1 score of 0.79. Finally, the paper published by Karthik Desingu et al. [15] employed a transfer learning method by ensembling the features extracted from Inception-ResNet-v2 with the metadata information after preprocessing the images.

Based on the results from last year's submissions, our team observed that 2 architectures stood out among the rest — ResNet101 and EfficientNetB0, producing very high individual F1 scores with base models. Also, most of the teams reported that using an ensemble model with gradient boosting and the usage of transfer learning algorithms gave very high F1 score and classification accuracy.

3. Methods

Transfer Learning [16] is a technique where a neural network is trained on a given problem domain, and then used on another similar problem by adding one or more layers to the trained model to fine-tune the model to solve the new problem at hand. This technique is generally used to reduce the training time for a neural network model and also results in a lower generalization error. The weights in the old model are typically used as a starting point in the fine-tuning process. Ensembling [17] is another popular computational learning technique, where the predictions or extracted features from multiple machine learning models are employed for classification or regression, aiming to achieve improved performance than that of a single model.

Our team applied transfer learning to calibrate state-of-the-art networks like ResNet101, ResNeXt101 and EfficientNets for snake species classification. Our proposed method extracts representative features from input snake images. The feature vectors were then fed to a gradient boosting ensemble classifier, along with contextual categorical features to predict the class probabilities for a given observation, thereby inferring its most probable class label.

3.1. Dataset

The given dataset consists of 187129 snake observations made with 318532 images belonging to 1572 unique species. A set of 208 countries are also recorded. The given metadata gives us useful geographical information such as the country where the snake was spotted. This can be used to supplement the visual information during training. The current dataset is vast compared to last year's challenge dataset which had only 772 distinct snake species on the contrary having a 414424 images. The number of images per species has drastically reduced from 536 last year to 212 this year.

While last year SnakeCLEF2021 [18] has 414424 photographs of 772 snake species collected in 188 countries where present in the dataset, this year featured a significantly lower number of photographs bringing the total down to 318532. The number of snake species incorporated this year was 1572, increasing the number of species by more than a factor of 2 compared to the previous years. The photographs were captured from across 208 countries, showing an increase of 10.60% compared to the previous year. The dataset is particularly challenging due to its long-tailed distribution — it has a heavy long-tailed class distribution, where the most frequent species — *Natrix natrix* is represented by 6472 images and the least frequent species by just 5 samples.

3.2. Input Image Preprocessing

The given images were first fed into an input sequencer where we noticed a lot of noisy data. Analysis of the images presented the fact that they differed in size and were of different scales. We thus employed a fold-down strategy to convert all images to the RGB format and resize them to a standard size of $224 \times 224 \times 3$ using bi-linear interpolation [19].

To eliminate the effect of irrelevant factors in the context of the required task such as variation in lighting conditions among the photographs, the images were linearly normalized to values between 0 and 1 Transformations such as scale and rotation, as well as contrast and saturation variations were induced on the model inputs to make the model more generic, immune to the impact of positional and orientation-based bias and prevent memorization by enhancing image diversity. RandAugment [20] was used to augment the input images using the aforementioned transformations. RandAugment is parameterized by two values - the number of augmentation transformations to apply sequentially (N), and the magnitude for all the transformations (M). The values used in [20] for the ResNet model i.e N=3 and M=4 were chosen, by example.

3.3. Feature Extraction

Reducing the number of resources to train on without losing relevant information is the basis on which feature extraction is built. The dimensions are reduced such that the raw data is made into chunks of manageable groups for processing. This method proved useful as the given large dataset had over 110 GB of training images and thus the amount of features that actually represent the dataset are shadowed by widely unrepresentative features.

Concretely, state-of-the-art neural networks were used as feature extractors that output representation vectors for the input images. The feature extraction process is tuned to extract representative and class-discriminate features through supervised learning, wherein class labels are used as ground truth to backpropagate [21] and tune the model weights.

3.4. Deep Learning Architectures Considered

Multiple deep-learning architectures were considered for this classification task.

ResNet101 [22] is a popular convolutional neural network model developed in 2015. This model solves the degradation problem which states that as the network depth increases accuracy gets saturated and then degrades rapidly. ResNet101 uses shortcut connections that skip one or more layers to solve the degradation problem which was inspired from the Highway network [23] which used gated shortcut connections to control the flow of information in the shortcut.

EfficientNet(s) [24] are a class of convolutional neural networks that were built in 2019. It's a small-scale architecture with about 11 million trainable parameters. It was created with the help of a multi-objective neural network that prioritized precision and floating point operations. It supports compound scaling while maintaining network balance across all dimensions. It employs an inverted bottleneck, as well as a depth-wise convolutional network that includes squeeze and excitation operations. It employs MBConv blocks [25] that serve as Inverted Linear BottleNeck layers. These layers use Depth-Wise Separable Convolution operations. The model complexities of the variants increase from B0 to B7. The authors experimented with B0, B4 and B6 variants to scale the model complexity and find the best suited intricacy for the dataset.

ResNeXt101 [26] is another popular convolutional neural network model which is very similar to the ResNet101 model. ResNet101 has many sequential layers whereas ResNeXt101 has parallel stacking layers instead. It follows a split-transform-merge strategy like the Inception module [27]. Unlike the Inception module which has different filters and sizes for each block, ResNeXt shares these hyper-parameters for all the blocks.

3.5. Gradient Boosting Ensemble Classifier

Ensemble methods [17] are techniques employed to combine multiple model(s) to produce improved results. They boast higher accuracy scores than the individual models themselves. Boosting [28] is a prominent ensembling technique used wherein new models are added to the existing features of the model to correct errors. Our solution adopted a gradient-boosting ensemble approach to classify images into their corresponding snake species.

XGBoost [29] is an implementation of gradient boosted decision trees designed for speed and performance. The XGBoost library package was chosen by the authors for implementation among all the other available boosters for its superior execution speed.

4. Experiments

Model training was performed through transfer-learning from the weights obtained upon training with the ImageNet data set [30], and fine-tuning on the SnakeCLEF-2022 training data. During model training, the models' prediction accuracy were tracked to later choose the set of feature extractors to use for ensembling. Based on the observed performance of each network, ResNeXt101 and EfficientNetB6 were chosen as feature extractors for ensembling. The XGBoost gradient boosting ensemble classifier was used for ensembling.

The forward propagation during training and prediction is described. Each observation is composed of multiple snake images along with its contextual geographic information — the country where the species was observed. Each image in an observation is first preprocessed, then passed through the two feature extraction networks to obtain two representation vectors, each of size 4096. These vectors are merged together, along with numeric encoded country metadata for the image, to obtain a final vector of size 8193. The numeric encoding was achieved with the help of Label Encoders from the scikit-learn [31] library wherein, the categorical country codes available as training data are converted to integral class labels that can be fed as input to the feature extraction neural network. These 8193 features are fed to the boosting ensemble classifier to obtain a probability distribution over all possible snake species classes. This workflow is depicted in Figure 1. Countries that are present in the testing data, but not encountered in the training set are encoded as 0, a commonly adopted strategy to deal with unseen categorical data in deep learning methods.

The corresponding class probability values obtained for each image in an observation are averaged to obtain a single aggregate distribution of probabilities over all classes. Consequently, there is one single probability distribution for each observation. The class that is attributed with the highest aggregate probability value is output as the classification label.

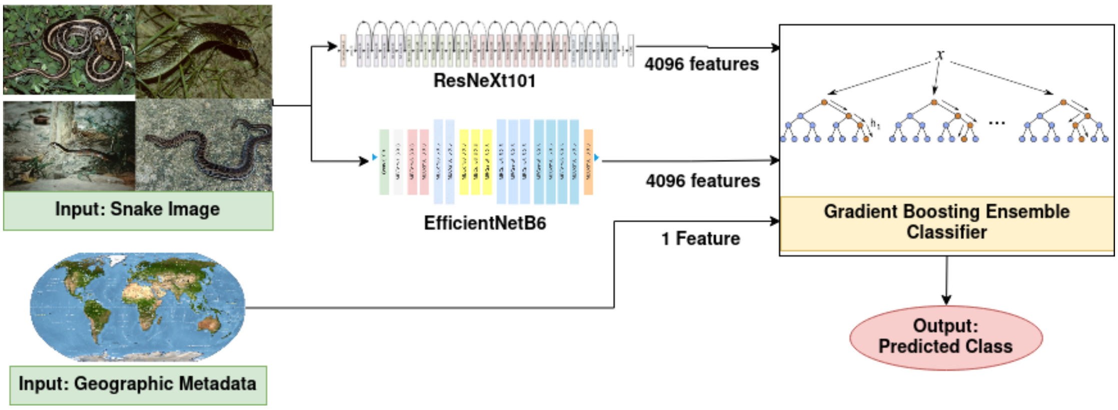

Figure 1: Prediction workflow used for the classification of snake species using an gradient boosted ensembling classifier. *Note*: Model architectures depicted are illustrative only and NOT accurate representations of the underlying network design.

4.1. Model Training

The details of the model training process, performed through transfer-learning is presented in this section. A summary of the parameters used for model training is tabulated in Table 1.

Table 1
Model training parameters used to train each of the convolutional neural networks used for this classification task.

Parameter	Optimizer	Learning rate	Batch size	Epochs
ResNet101	Adam	$3e{-}4$	32	40
EfficientNetB0	Adam	$1e{-}5$	32	40
EfficientNetB4	Adam	$1e{-}4$	32	40
EfficientNetB6	Adam	$3e{-}2$	32	40
ResNeXt101	Adam	$1e{-}3$	32	30

4.1.1. ResNet101

The feature extraction layers of ResNet101 were trained with a two-step classification block, comprising two dense blocks with 4096 and 1572 neurons respectively. The extracted features were percolated through a flatten layer to obtain, before feeding to the classification block. In addition, a dropout layer was added after the dense layer to avoid overfitting. Dropout rates between 0.30 and 0.70 were experimented and set to 0.40 in the final version of the model. The model was trained with the Adam optimizer at an initial learning rate of $3e{-}4$. It was backpropagated using the Categorical Cross-Entropy (CCE) loss. For feature extraction, the output of the first dense layer was used to produce a feature vector of 4096 elements. During training, the model's prediction accuracy was tracked to later choose the feature extractor to use for ensembling.

4.1.2. EfficientNetB0

EfficientNetB0 was also trained with a two-step classification block, comprising two dense blocks with 4096 and 1572 neurons respectively. The extracted features were percolated through a flatten layer to obtain, before feeding to the classification block. The dropout layer added after the dense layer for this network was experimented between 0.30 and 0.70 and fixed at 0.45. The model was trained with the using the Adam optimizer at an initial learning rate of $1e{-}5$. It was back propagated using the Categorical Cross-Entropy (CCE) loss. For feature extraction, the output of the first dense layer was used to produce a feature vector of 4096 elements.

4.1.3. EfficientNetB4

EfficientNetB4 was trained with a two-step classification block, also comprising two dense blocks with 4096 and 1572 neurons respectively. While the final layers of the architecture are same as the other EfficientNet models, the dropout layer added after the dense layer for this network was experimented between 0.30 and 0.70 and fixed at 0.35 for this model. The model was trained using the Adam optimizer at an initial learning rate of $1e{-}4$. Categorical Cross-Entropy (CCE) loss was used to back-propagate, and a 4096-sized vector was extracted as feature-representation.

4.1.4. EfficientNetB6

EfficientNetB6 followed the same approach — two dense blocks with 4096 and 1572 neurons, dropout layers, and CCE loss for back-propagation, and optimized with Adam — with differences only in the hyperparameters. The dropout layer after the dense layer was fixed at a dropout rate of 0.45 after experiments. The initial learning rate was set at $3e{-}2$. A 4096-sized vector was extracted as feature-representation.

4.1.5. ResNeXt101

The ResNeXt101 architecture was augmented following the same strategy as the aforementioned two i.e. by adding a two-step classification block of 4096 and 1572 neurons respectively. Here, the dropout after experimenting, was set at 0.30. An Adam optimizer along with CCE loss was used for training, with an initial learning rate of $1e{-}3$.

4.2. Loss, Metrics, Activation and Optimizer Used

The specific details of the loss functions, activation functions, optimizer, and evaluation metrics used during the experiments are laid out in this section.

4.2.1. Adam Optimizer

Adam [32] is a stochastic optimization method which is used on gradient descent and maintains a single learning rate (alpha) throughout training. Adam combines the advantages of the Adaptive Gradient Algorithm and Root Mean Square Propagation. Unlike the Root Mean Square Propagation in which the first moment about the mean is used, Adam uses the average of the second moments about the mean too. In effect, Adam provides an optimization algorithm that can handle sparse gradients on noisy problems, by maintaining a per-parameter learning rate.

4.2.2. Categorical Cross Entropy Loss

The categorical cross entropy is a measure of the difference between two discrete probability distributions. It is calculated using the formula in Equation 1.

$$Loss = -\sum_{i=1}^{n} y_i log t_i, \tag{1}$$

where, y_i represents the corresponding target value for t_i the scalar model output.

4.2.3. Softmax Activation

The Softmax activation function is used at the end of the output layer to produce the posterior probability distribution over all classes, based on equation 2. Softmax is essentially a mathematical function that converts a vector of numbers into a vector of probabilities, where the probabilities of each value are proportional to the relative scale of each value in the vector. In effect, it normalizes the outputs, converting them from weighted sum values into probabilities that sum to one.

$$softmax(z_i) = \frac{exp(z_i)}{\Sigma_j exp(z_j)} \tag{2}$$

where, z represents the values from the neurons of the output layer. The exponential acts as the non-linear function. These values are divided by the sum of exponential values to normalize and convert them into probabilities.

4.2.4. F1-Score Metric

The F1-Score is usually calculated as the harmonic mean of precision and recall. This is concretely expressed in Equation 3.

$$F_1 = \frac{2p_s r_s}{p_s + r_s}, \tag{3}$$

$$p_s = \frac{T_p}{T_p + F_p} \tag{4}$$

$$r_s = \frac{T_p}{T_p + F_n} \tag{5}$$

where, F_1 represents the F1-score, p_s represents precision, r_s represents recall, T_p represents true-positive, F_p represents false-positive and F_n represents false-negative. The contest prescribed *macro-averaged F1-Score* as the evaluation metric.

4.2.5. Accuracy Metric

The accuracy score (Acc) is computed as the ratio of correct predictions to the total number samples. This is expressed in Equation 6.

$$Acc = \frac{T_p + T_n}{T_p + T_n + F_p + F_n} \tag{6}$$

where, Acc represents the accuracy score, T_p represents true-positive, T_n represents true-negative, F_p represents false-positive and F_n represents false-negative.

4.3. XGBoost Ensemble Classifier

The XGBoost classifier was used for ensembling. While using XGBoost, hyperparameters were tuned [33] for optimal performance by trial and error. The maximum depth of the tree was set to 32. Increasing this value would make the model more complex and prone to overfitting. Increasing this value would also aggressively consume memory while training the deep tree and thus a lower value of 32 was set.

It was observed that learning rates higher than 0.10 lead to quick divergence, hence values in the range of $10e{-}3$ to $10e{-}5$ were used. Grid-search was performed by varying the learning rates in this range, with decision trees in the range of 100 to 1000. Combinations having the least losses were chosen to further tune the tree-level parameters. The maximum depth for the tree is left to be determined based on the training progress of the classifier and is not set strictly. This causes the depth to expand until the leaves are pure (has all samples belonging to the same class) or has reached the threshold of minimum number of samples required to split further. Due to the long-tailed distribution of the data set, some classes may require deeper branches to capture more information from the features. To control overfitting, an upper limit was set on the number of leaves by performing a grid search over values in range of 32 to 256.

Then, objective parameters helps specify the learning task and corresponding learning objective. Softmax was chosen as the objective function. The classifier was configured for multiclass classification and the number of classes was explicitly set to 1572.

5. Results and Conclusion

The country metadata was used as a categorical feature in the ensemble classifier. It showed a strong impact on the classification result. With inclusion of this categorical country metadata, the testing F1-score of the best submission improved from 2.68% to 3.64%. Likewise, the cross-validation accuracy and F1-score for the same model improved from 37.62% and 26.55% to 44.91% and 29.31%, upon inclusion of the geographic data.

Figure 2 represents the relative importance of the 20 most effective image or metadata features of the 20 most impactful features used to train the ensemble classifier. The feature importance values were normalized and scaled between 0 and 100 to realize the relative impacts. Features named as f1, f2, etc. denote features extracted from the neural networks It is worth mentioning that f0 through f4095 denote features extracted using EfficientNetB0, while f4096 through

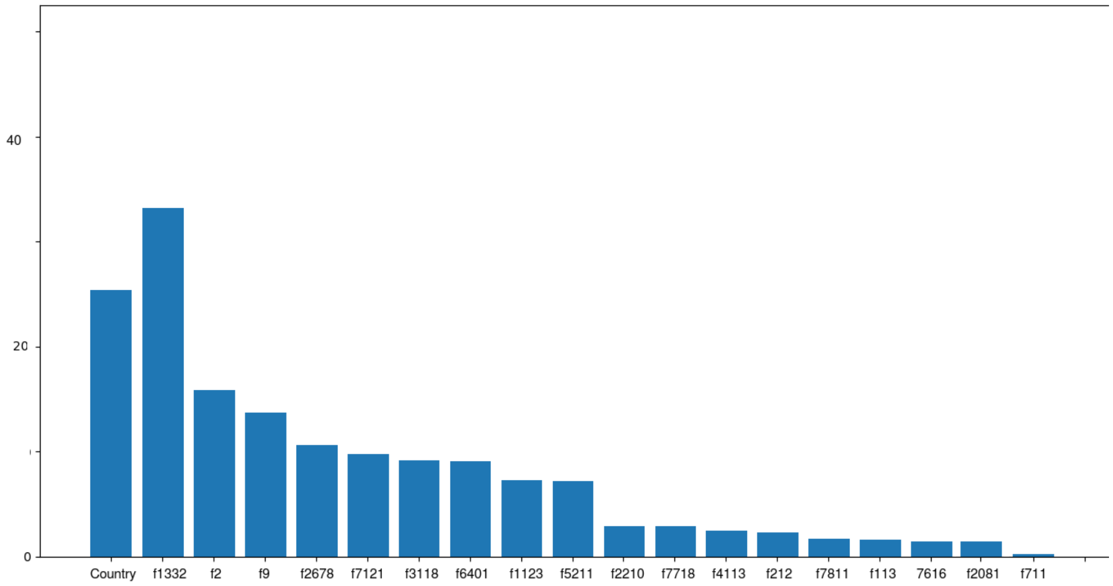

Figure 2: Relative importance on a scale of 0-100 of the 20 most impactful features used to train the classifier. The first bar represents feature importance of country feature.

f8191 represent the ResNet101-extracted features. It is evident that country information has a signification influence on classification.

After deciding the baseline architectures for ensembling — namely, EfficientNetB6 and ResNeXt101 — based on individual prediction performance, the baseline models were trained towards convergence. Following this, the multiple ensemble models were trained using the gradient boosting classifier using the two baseline models and contextual data, with different hyperparameter settings. The ensemble classifier's performance was improved over several runs, by tuning the hyperparameters of the gradient boosting classifier. The contest prescribed F1-scores macro-averaged across all classes as the evaluation metric. Model runs were evaluated on the given stratum of validation set using this metric. The metrics were evaluated as an average over the five iterations (for 5-fold cross validation) performed in each run during training.

The top scoring models were then used to perform prediction on the test data and evaluated on the contest website. The top-5 results from this pool of predictions is summarized in Table 2 for competition submissions, and in Table 3 for post-competition evaluations.

Our team achieved a training accuracy of 45.78%, validation accuracy of 44.91%. The corresponding model secured an F1-score of 3.64% on the competition's test data. Our team placed 38th among 51 participating teams. These results are summarized in Table 2 below:

The results depict the integration of contextual geographic data for snake species classification in a positive light. Furthermore, the fine-tuning and ensembling of features extracted using multiple neural architectures, and merging contextual data looks promising. Several existing

Table 2
Performance metrics of the 5 best submissions. F1-Scores are macro-averaged across classes.

Submission#	Validation Accuracy	Validation F1-Score	Test F1-Score
1	44.91	29.31	3.64
2	40.72	26.13	3.38
3	35.11	25.27	2.53
4	35.18	25.46	2.15
5	34.97	22.18	2.05

approaches have introduced metadata such as population counts of various species, more location-specific geographic data such as city, state and climatic features such as temperature and humidity. An interesting approach is to employ class-wise probability priors to the neural networks based on such metadata [34].

On account of insufficient computing resources to complete all model training experiments in time for the SnakeCLEF's large snake dataset, results were submitted before complete model convergence. Post the deadline, significant improvements were observed in classification accuracy, particularly with Submission Numbers 1 and 4 (refer to Table 2) were observed upon further training the baseline neural networks, and tuning hyperparameters of the gradient boosting ensemble classifier. A summary of post-competition improvements in prediction results during the working notes submission phase of SnakeCLEF-2022 is tabulated in Table 3.

Table 3
Performance metrics of the 5 best submissions **post-competition**, ordered best to worst. F1-Scores are macro-averaged across classes.

Submission#	Training Accuracy	Validation Accuracy	Validation F1-Score	Test F1-Score
1	89.11	87.13	52.04	51.39
2	86.01	85.39	51.42	49.94
3	80.86	80.44	46.13	46.11
4	79.81	78.92	47.19	45.52
5	78.64	77.14	44.11	42.18

Our best post-competition result is at par with the 11^{th} best of the 51 contesting teams. It is apparent that subsequent hyperparameter tuning and training of the baseline networks, as well as the boosting classifier, have been effective in improving the model performance. Hence, the team believes and strongly advocates that for data-intensive and high-complexity image classification tasks that are commonly released as LifeCLEF tasks, the adopted ensembling approach using gradient boosting is an impactful option. We further conjecture that training the individual models to convergence, and subsequently applying the boosting ensembler with hyperparameter tuning will culminate in a far more superior classifier performance, that exhausts the enitrety of the proposed architectures' and methodology's potential. Furthermore, approaches involving input image resolution variations, usage of alternative pre-trained weights [35], as well as the inclusion of custom training layers to the frozen base model when transfer learning [36] can greatly improve the quality of feature extraction. In this participation, for reasons of time constraints already described, the application of image preprocessing techniques could not be experimented with exhaustively and systematically. The authors further speculate that such an exercise would be of significance in improving the overall model performance, particularly that of the baseline neural networks used for feature extraction [37].

Other approaches, such as varying input image resolutions and employing alternative pre-trained weights [35] as well as including custom training layers to the frozen base model, in addition to the classification block [36], can improve classifier performance. Finally, the application of image preprocessing techniques can play a significant role in improving the feature extraction ability of convolutional neural networks [37].

Acknowledgments

The authors would like to express their gratitude to the Machine Learning Research Group (MLRG), Department of Computer Science and Engineering, Sri Sivasubramaniya Nadar College of Engineering, Chennai, India (https://www.ssn.edu.in/) for providing the GPU resources for model training and testing.

References

[1] L. Picek, A. M. Durso, M. Hrúz, I. Bolon, Overview of SnakeCLEF 2022: Automated snake species identification on a global scale, in: Working Notes of CLEF 2022 - Conference and Labs of the Evaluation Forum, 2022.

[2] A. Joly, H. Goëau, S. Kahl, L. Picek, T. Lorieul, E. Cole, B. Deneu, M. Servajean, A. Durso, H. Glotin, R. Planqué, W.-P. Vellinga, A. Navine, H. Klinck, T. Denton, I. Eggel, P. Bonnet, M. Šulc, M. Hruz, Overview of lifeclef 2022: an evaluation of machine-learning based species identification and species distribution prediction, in: International Conference of the Cross-Language Evaluation Forum for European Languages, Springer, 2022.

[3] A. Joly, H. Goëau, S. Kahl, L. Picek, T. Lorieul, E. Cole, B. Deneu, M. Servajean, A. Durso, I. Bolon, et al., Lifeclef 2022 teaser: An evaluation of machine-learning based species identification and species distribution prediction, in: European Conference on Information Retrieval, Springer, 2022, pp. 390–399.

[4] B. Mohapatra, D. A. Warrell, W. Suraweera, P. Bhatia, N. Dhingra, R. M. Jotkar, P. S. Rodriguez, K. Mishra, R. Whitaker, P. Jha, for the Million Death Study Collaborators, Snakebite mortality in india: A nationally representative mortality survey, PLOS Neglected Tropical Diseases 5 (2011) 1–8. URL: https://doi.org/10.1371/journal.pntd.0001018. doi:10.1371/journal.pntd.0001018.

[5] Z. Yang, R. Sinnott, Snake detection and classification using deep learning (2021).

[6] J. M. Gutiérrez, J. J. Calvete, A. G. Habib, R. A. Harrison, D. J. Williams, D. A. Warrell, Snakebite envenoming, Nature reviews Disease primers 3 (2017) 1–21.

[7] L. Bloch, C. M. Friedrich, Efficientnets and vision transformers for snake species identification using image and location information, in: G. Faggioli, N. Ferro, A. Joly, M. Maistro, F. Piroi (Eds.), Proceedings of the Working Notes of CLEF 2021 - Conference and Labs of the Evaluation Forum, Bucharest, Romania, September 21st - to - 24th, 2021, volume 2936 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021, pp. 1477–1498.

[8] L. Kalinathan, P. Balasundaram, P. Ganesh, S. S. Bathala, R. K. Mukesh, Automatic snake classification using deep learning algorithm, in: CLEF, 2021.

[9] A. James, Snake classification from images, 2017. doi:10.7287/peerj.preprints.2867.

[10] M. Rajabizadeh, M. Rezghi, A comparative study on image-based snake identification using machine learning, Scientific reports 11 (2021) 1–16.

[11] V. Aggarwal, et al., A review: deep learning technique for image classification, ACCENTS transactions on image processing and computer vision 4 (2018) 21.

[12] R. Borsodi, D. Papp, Incorporation of object detection models and location data into snake species classification (2021).

[13] R. Chamidullin, M. Šulc, J. Matas, L. Picek, A deep learning method for visual recognition of snake species (2021).

[14] L. G. Coca, A. T. Popa, R. C. Croitoru, L. P. Bejan, A. Iftene, Uaic-ai at snakeclef 2021: Impact of convolutions in snake species recognition (2021).

[15] K. Desingu, M. Palaniappan, J. Kumar, Snake species classification using transfer learning (2021).

[16] L. Torrey, J. Shavlik, Transfer learning, in: Handbook of research on machine learning applications and trends: algorithms, methods, and techniques, IGI global, 2010, pp. 242–264.

[17] Z.-H. Zhou, J. Wu, W. Tang, Ensembling neural networks: many could be better than all, Artificial intelligence 137 (2002) 239–263.

[18] L. Picek, A. M. Durso, I. Bolon, R. R. de Castañeda, Overview of snakeclef 2021: Automatic snake species identification with country-level focus (2021).

[19] P. Smith, Bilinear interpolation of digital images, Ultramicroscopy 6 (1981) 201–204.

[20] E. D. Cubuk, B. Zoph, J. Shlens, Q. V. Le, Randaugment: Practical automated data augmentation with a reduced search space, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020, pp. 702–703.

[21] D. E. Rumelhart, R. Durbin, R. Golden, Y. Chauvin, Backpropagation: The basic theory, Backpropagation: Theory, architectures and applications (1995) 1–34.

[22] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, 2015. URL: https://arxiv.org/abs/1512.03385. doi:10.48550/ARXIV.1512.03385.

[23] R. K. Srivastava, K. Greff, J. Schmidhuber, Training very deep networks, Advances in neural information processing systems 28 (2015).

[24] M. Tan, Q. V. Le, Efficientnet: Rethinking model scaling for convolutional neural networks (2019). URL: https://arxiv.org/abs/1905.11946. doi:10.48550/ARXIV.1905.11946.

[25] M. Tan, Q. Le, Efficientnetv2: Smaller models and faster training, in: International Conference on Machine Learning, PMLR, 2021, pp. 10096–10106.

[26] S. Xie, R. Girshick, P. Dollár, Z. Tu, K. He, Aggregated residual transformations for deep neural networks, 2016. URL: https://arxiv.org/abs/1611.05431. doi:10.48550/ARXIV.1611.05431.

[27] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, 2014. URL: https://arxiv.org/abs/1409.4842. doi:10.48550/ARXIV.1409.4842.

[28] A. Natekin, A. Knoll, Gradient boosting machines, a tutorial, Frontiers in neurorobotics 7 (2013) 21.

[29] T. Chen, C. Guestrin, XGBoost, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016. URL: https://doi.org/10.1145%2F2939672.2939785. doi:10.1145/2939672.2939785.

[30] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE conference on computer vision and pattern recognition, Ieee, 2009, pp. 248–255.

[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, the Journal of machine Learning research 12 (2011) 2825–2830.

[32] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014. URL: https://arxiv.org/abs/1412.6980. doi:10.48550/ARXIV.1412.6980.

[33] A. Anghel, N. Papandreou, T. Parnell, A. De Palma, H. Pozidis, Benchmarking and optimization of gradient boosting decision tree algorithms, arXiv preprint arXiv:1809.04559 (2018).

[34] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang, W. Xu, Cnn-rnn: A unified framework for multi-label image classification, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 2285–2294.

[35] A. Joly, H. Goëau, S. Kahl, B. Deneu, M. Servajean, E. Cole, L. Picek, R. Ruiz de Castañeda, I. Bolon, A. Durso, et al., Overview of lifeclef 2020: a system-oriented evaluation of automated species identification and species distribution prediction, in: International Conference of the Cross-Language Evaluation Forum for European Languages, Springer, 2020, pp. 342–363.

[36] M. Zhong, J. LeBien, M. Campos-Cerqueira, R. Dodhia, J. L. Ferres, J. P. Velev, T. M. Aide, Multispecies bioacoustic classification using transfer learning of deep convolutional neural networks with pseudo-labeling, Applied Acoustics 166 (2020) 107375.

[37] K. K. Pal, K. Sudeep, Preprocessing for image classification by convolutional neural networks, in: 2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), IEEE, 2016, pp. 1778–1781.